OOP – ASSIGNMENT 2

Q.1

Customer

Customer class will initialise information such as the customer's first name, surname, ID and email address. The Shopping Cart class will be dependent on this class as a Cart cannot exist without a Customer to belong to. A random number will have to be generated using the Math.random() method for the Customer ID. It also has a relationship with order so its objects can be used in email and transaction test.

Shopping Cart

The Shopping Cart class should take objects from the Item class and make Array lists with these. Functionality should include methods which add items, remove items, close the cart such that no other items should be added and clear the cart. It should also list all the items and total the prices of all items. Boolean checks will be necessary to see if functionality can be accessed depending on whether the cart is open or not. For loops with indexing will be needed for displaying lists and totaling price values. It also has a relationship with order so its objects can be used in email and transaction test.

Order

Shopping Cart objects are to be transferred to this class one by one. It should also take relationships from other classes such as Customer, Shopping Cart, Address and Email. Math.random() should also be used to generate an Order number.

Item

Found in lecture slides. Usual get and set methods. It also has a relationship with order so its objects can be used in email and transaction test.

Payment

The Payment class will have information such as the card number, expiration date, card type. It will also verify its own objects using a Boolean field and an isValid() method. This class will also be dependent on the Order class for most pieces of information in the order object. It must verify that the cardnumber, expiration date and card type are valid and have an isValid() method for use in the transaction test.

Email

This should print two messages depending on the scenarios of the order failing or the order being successful. This class is also dependent on the order class for printing information from Customer, Cart, Address, etc. It has a relationship with order so its objects can be used in email and transaction test.

Address

This class should have get and set methods for door number, address line and Eircode. It also has a relationship with order so its objects can be used in email and transaction test.

Q.2

CUSTOMER CLASS

```
public class Customer

{


    private String firstName;

    private String surName;

    private String emailAddress;

    private final long customerId;
}



public Customer(String firstName, String surName, String emailAddress){
this.firstName = firstName;
this.surName = surName;
this.emailAddress = emailAddress;
customerId = makeCustomerId();
}

public long getId(){
    return customerId;
}

public String getName(){
   return surName;
}

public String getEmail(){
   return emailAddress;
}

private long makeCustomerId()
{
   return (long)(Math.random() * 999999 + 100000);


}
```

```
}

ADDRESS CLASS

public class Address{

  private int doorNum;
  private String addLine;
  private String eirCode;

public Address(int doorNum, String addLine, String eirCode)
{
  this.doorNum = doorNum;
  this.addLine = addLine;
  this.eirCode = eirCode;
}


public int getDoorNum()
{
   return doorNum;
}

public void setDoorNum(int dNum)
{
   doorNum = dNum;
}

public String getAddLine()
{
  return addLine;
}

public void setAddLine(String aLine)
{
 addLine = aLine;
}

public String getEirCode()
{
   return eirCode;
}

public void setEirCode(String eCode)
{
   eirCode = eCode;
}


}
```

PAYMENT CLASS

```
public class Payment
```

```
{
   private int expiryYear;
   private int expiryMonth;
   private String cardType;
   private long cardNum;
   private boolean valid;

   public Payment(Order order, String cardType, long cardNum, int expiryMonth, int
expiryYear)
   {
      this.cardType = cardType;
      this.valid = valid;
   }

   private String cardNumStr = Long.toString(cardNum);

   public void Valid(){

   if (cardNumStr.length()== 16 || expiryMonth>10 && expiryYear>=24 ||
cardType=="MasterCard" || cardType=="Visa" )
   {
      valid = true;
      System.out.println("Card details successfully validated\n");
   }
   else
    {valid = false;
```

```
     System.out.println("Card details invalid, please try again\n");
    }
}



    public boolean isValid()
    {
        return valid;
    }

 }
```

ITEM CLASS

```
public class Item
{
    private String name;
    private double price;
    private int itemId;



    public Item(String itemName, int Id){
        name = itemName;
        itemId = Id;
    }

    public void setPrice(int price){
      this.price = price;
    }

    public double getPrice(){
      return price;
    }

    @Override
    public String toString(){
      String out = "itemID: " + itemId + "  " + name + "  €" + String.format("%.2f", price) + "\n";
      return out;
    }




}
```

SHOPPINGCART CLASS

```
import java.util.ArrayList;

public class ShoppingCart
{
 private double totalPrice;
 private boolean closed;
 public ArrayList<Item> list;
 public ShoppingCart(Customer customer)
 {
    list = new ArrayList<>();
 }

 public void addItem(Item s)
 {
   if(!closed)
   {list.add(s);
   }
   else{
   System.out.println("Sorry the Shopping Cart is closed");
   }
}

public void removeItem(int index)
   {
      if(!closed && list.get(index)!=null)
      {
         list.remove(index);
      }
      else
      {
         System.out.println("Sorry the Shopping Cart is closed");
      }
   }

public Item getItem(int index){
  if(list.get(index)!=null)
  {
      return list.get(index);
  }
  else
  {
     System.out.println("This item object does not exist");
     return null;
  }
}

public int numItems()
{
   return list.size();
}

public void listItem()
{
   for(int i=0; i<numItems(); i++)
   {
       System.out.println(list.get(i).toString());
   }
}
```

```
//totals the prices of the items
public void totalPrItem()
{
  for(int i=0; i<numItems(); i++)
  {
    totalPrice = totalPrice + getItem(i).getPrice();
  }

  System.out.println("Total Price :€" + String.format("%.2f", totalPrice));
}

public double getTotal()
{
  return totalPrice;
}

 public void closed(){
    closed = true;
    System.out.println("The Shopping Cart is now closed");
  }

 public void clear (){
    list.clear();
  }
```

```
}
```

ORDER CLASS

```
import java.util.ArrayList;

public class Order
{

   public Customer customer;
   private ShoppingCart cart;
   private Address address;
   private Address deliveryAddress;
   private ArrayList<Item> orderList;
   public long orderNum;


  public Order(Customer customer, ShoppingCart cart, Address address)
  {
     this.customer = customer;
     this.cart = cart;
     this.address = address;
   orderNum = makeOrderNo();
   orderList = new ArrayList<>();
  }
```

```
//generates random order number
 private long makeOrderNo()
 {
    return (long)(Math.random() * 999999 + 100000);


 }


public long getorderNum(){
   return orderNum;
}


public String getNCustomer() {
    return customer.getName();
 }


 public String getECustomer(){
    return customer.getEmail();
 }


 public int getNoItems(){
    return cart.numItems();
 }


 public double getTotalP(){
    return cart.getTotal();
   }



  public ShoppingCart getShoppingCart(){

     return cart;

  }


  public Address getAddress(){

     return address;

  }


}
```

EMAIL CLASS

```
public class Email

{

   private Order order;


public Email(Order order)

{

   this.order = order;

}



   public void orderSuccess()

   {
        System.out.println("\nTo " + order.getECustomer()+   "\nOrder NO." +
order.getorderNum()+ "\n Dear " + order.getNCustomer()+"," + "\n Your Order has been
successfully processed");

        System.out.println("\nSummary; Items ordered:"+ order.getNoItems()+ ", Total Amount
Paid €" + order.getTotalP());

        System.out.println("\nItems will be shipped to: \n" +order.getAddress().getDoorNum()+
" "+ order.getAddress().getAddLine()+"\n" + order.getAddress().getEirCode() + " in 3-5
business days");

    }


    public void orderFailure()

    {

        System.out.println("\nTo " + order.getECustomer()+   "\nOrder NO." +
order.getorderNum()+ "\n Dear " + order.getNCustomer()+"," + "\n Unfortunately your order
has failed to be processed");



    }




}
```

TRANSACTION TEST CLASS

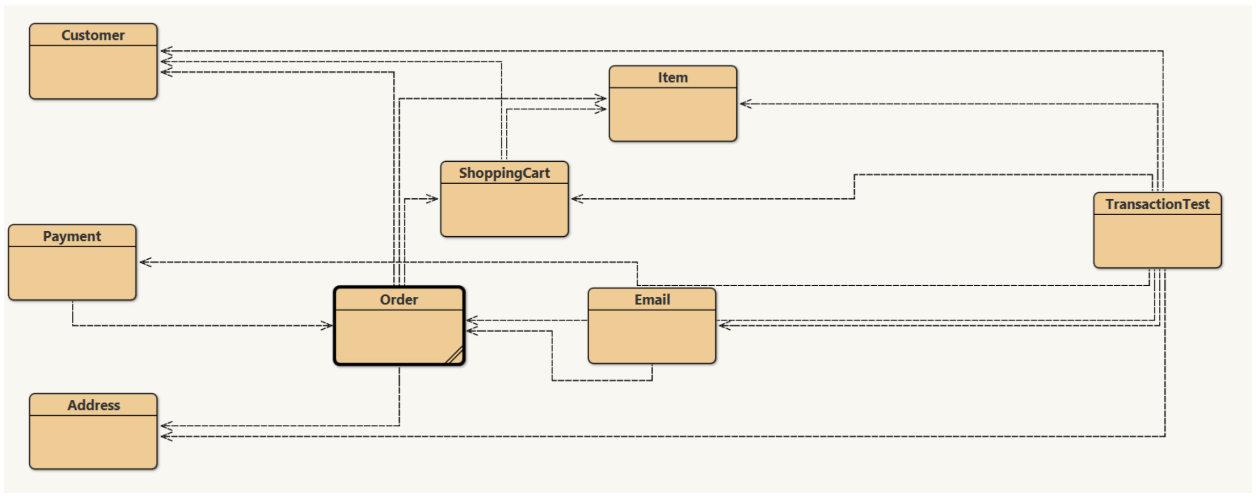
```
public class TransactionTest

{
```

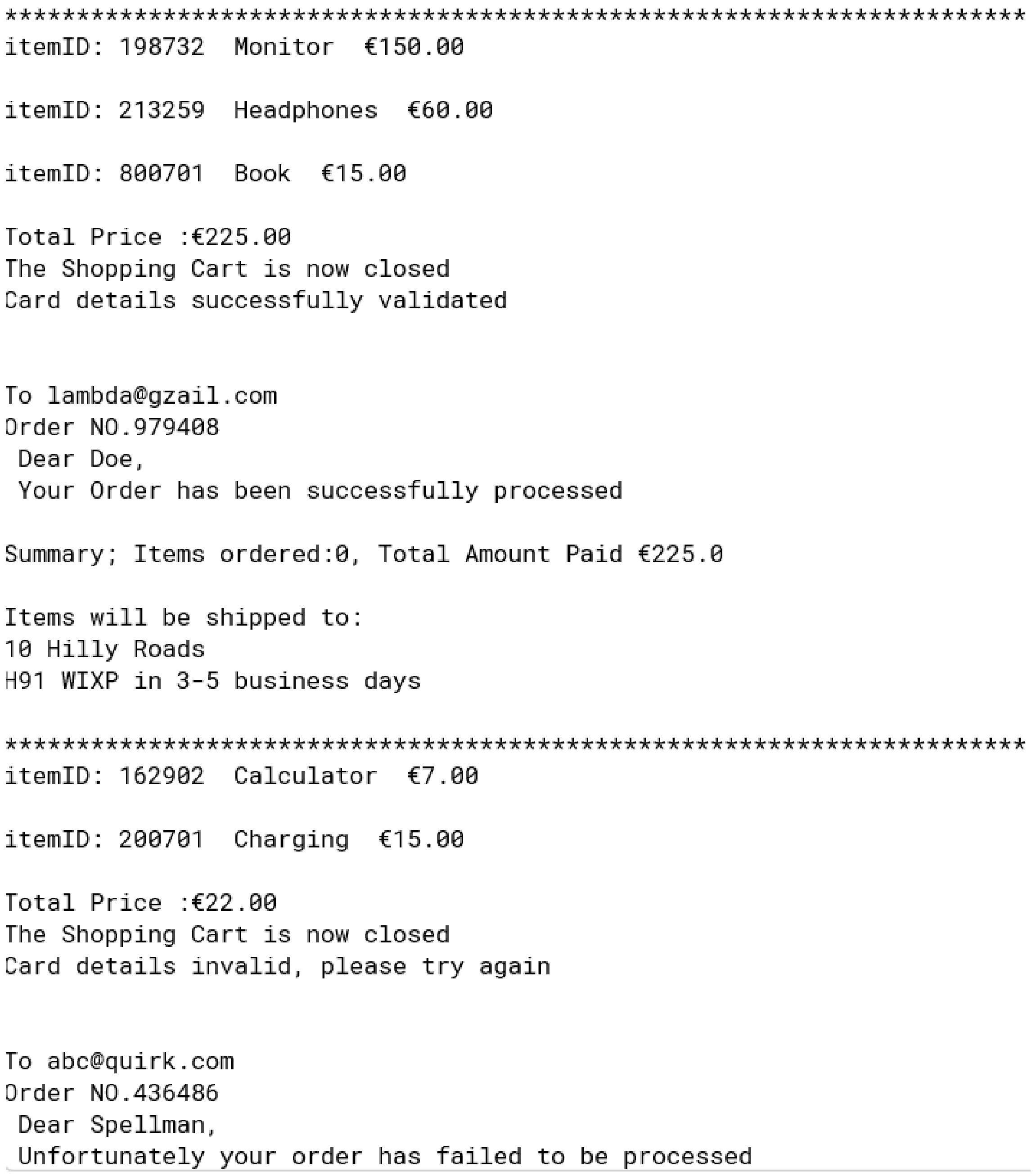
```
public TransactionTest()
{

}



public static void main(String[] args)
{
 TransactionTest test = new TransactionTest();
 test.transaction1();
 test.transaction2();
 test.transaction3();

}



public void transaction1(){
   Item item1 = new Item("Monitor", 198732);
   Item item2 = new Item("Headphones",213259);
   Item item3 = new Item("Book", 800701);



   Customer customer = new Customer("John", "Doe", "lambda@gzail.com");
   ShoppingCart cart = new ShoppingCart(customer);
   Address address = new Address(10, "Hilly Roads", "H91 WIXP");
```

```
System.out.println("\n***************************************************************
*******");


  cart.list.add(item1);

  cart.list.add(item2);

  cart.list.add(item3);



  item1.setPrice(150);

  item2.setPrice(60);

  item3.setPrice(15);


  cart.numItems();

  cart.listItem();

  cart.totalPrItem();

  cart.closed();


  Order order = new Order(customer, cart, address);

  cart.clear();


  Payment payment = new Payment(order, "MasterCard", 1234567891234567L, 8, 25);


  payment.Valid();
```

```
   Email email = new Email(order);

   if (payment.isValid()){

     email.orderSuccess();

   }

   else{

      email.orderFailure();

   }

}



public void transaction2(){

   Item item1 = new Item("Calculator", 162902);

   Item item2 = new Item("Keyboard",647328);

   Item item3 = new Item("Charging", 200701);



   Customer customer = new Customer("Keelan", "Spellman", "abc@quirk.com");

   ShoppingCart cart = new ShoppingCart(customer);

   Address address = new Address(15, "Salty Springs", "H91 D3P0");



System.out.println("\n****************************************************************
*******");


   cart.list.add(item1);

   cart.list.add(item2);
   cart.list.add(item3);



   item1.setPrice(7);

   item2.setPrice(45);

   item3.setPrice(15);

   //removing first item in cart

   cart.removeItem(1);

   cart.numItems();

   cart.listItem();

   cart.totalPrItem();

   cart.closed();

   Order order = new Order(customer, cart, address);

   cart.clear();

   //user inputs payment information however card type is incorrect

   Payment payment = new Payment(order, "MisterCard", 9034564597623567L, 12, 26);

   payment.Valid();

   Email email = new Email(order);

   if (payment.isValid()){

     email.orderSuccess();
   }
   else{
     email.orderFailure();
   }

 }

 public void transaction3(){

 }


 }
```

Q.3





```
**********************************************************************
itemID: 198732  Monitor  €150.00

itemID: 213259  Headphones  €60.00

itemID: 800701  Book  €15.00

Total Price :€225.00
The Shopping Cart is now closed
Card details successfully validated


To lambda@gzail.com
Order NO.979408
 Dear Doe,
 Your Order has been successfully processed

Summary; Items ordered:0, Total Amount Paid €225.0

Items will be shipped to:
10 Hilly Roads
H91 WIXP in 3-5 business days

**********************************************************************
itemID: 162902  Calculator  €7.00

itemID: 200701  Charging  €15.00

Total Price :€22.00
The Shopping Cart is now closed
Card details invalid, please try again


To abc@quirk.com
Order NO.436486
 Dear Spellman,
 Unfortunately your order has failed to be processed
```

Unfortunately number of items: did not work as intended.